2. $f(n) = n^2 + 35n - 750$, $g(n) = (73n - 39)(21n + 11)$ 0.5		
a) $f(n) \in O(g(n))$	b) $f(n) \in \Omega(g(n))$	c) $f(n) \in \Theta(g(n))$

3. $f(n) = 10$, $g(n) = \log 10$ 0.5		
a) $f(n) \in O(g(n))$	b) $f(n) \in \Omega(g(n))$	c) $f(n) \in \Theta(g(n))$

4. $f(n) = n \times (n-1)!$, $g(n) = \pi \times 10^n$ 0.5		
a) $f(n) \in O(g(n))$	b) $f(n) \in \Omega(g(n))$	c) $f(n) \in \Theta(g(n))$

```
for (int i = 1, i <= (n^3), i++)
    for (int j = 1, j <= (n/10), i++)
        print("****") // this expression costs 1
for (int k = 1, k <= square root(n), k++)
    function print("****") // this expression costs 10
```

$$\sum_{k=1}^{\sqrt{n}} 10 + \sum_{i=1}^{n^3} \square \sum_{j=1}^{\frac{n}{10}} 1 \qquad 1.5$$

$$= 10\sqrt{n} + \sum_{i=1}^{n^3} \square \sum_{j=1}^{\frac{n}{10}} 1 \qquad 0.5$$

$$= 10\sqrt{n} + \frac{1}{10}\sum_{i=1}^{n^3} n \qquad 0.5$$

$$= 10\sqrt{n} + \frac{1}{10}\, n \times n^3 \qquad 0.5$$

Question 2 [_____/6 Points] Find the asymptotic behavior of the following segments of algorithms, using sigma notation ($\sum$).

Segment# 1:

```
for (int i = 1, i <= n, i++)
    for (int j = 1, j <= i, j++)
        print("***") // this expression costs 1
```

I

If the number of disks n is the input size, choose the number of moves $M(n)$ required to solve the problem, given the following conditions:

1. The first rule "Only one disk can be moved at a time" is ignored, and the final disks' shape is not necessarily like the initial shape. 0.5			
a) $M(n) = 1$	b) $M(n) = n$	c) $M(n) = 2n$	d) $M(n) = n^2 - 1$

2. The second rule "It is forbidden to place a larger disk on top of a smaller one" is ignored, and the final disks' shape is not necessarily like the initial shape. 0.5

1. The first rule "Only one disk can be moved at a time" is ignored, and the final disks' shape is not necessarily like the initial shape. 0.5

a) $M(n) = 1$	b) $M(n) = n$	c) $M(n) = 2n$	d) $M(n) = n^{3-1}$

2. The second rule "It is forbidden to place a larger disk on top of a smaller one" is ignored, and the final disks' shape is not necessarily like the initial shape. 0.5

a) $M(n) = 1$	b) $M(n) = n$	c) $M(n) = 2n$	d) $M(n) = n^{3-1}$

Part 2 [4 Points]

Question 4 [_____/4 Points] Consider the Graph instance $G = (V, E), V = \{A, G, H, I\}$, and set of edges E having weights as defined by the weight matrix next.

4.1.[/3 Points] Trace the execution of the **exhaustive search approach** for the travelling salesman problem described in class on the given instance **assuming that the starting city is** A. Show the details and arrange your answer in a table.

	A	G	H	I
A	∞	7500	5000	2800
G	7500	∞	3000	6000
H	5000	3000	∞	4000
I	2800	6000	4000	∞

4.2.[_____/0.5 Points] What is the cost of the shortest Hamiltonian circuit?

4.3.[_____/0.5 Points] If the exhaustive search algorithm for TSP is programmed on a computer that executes ten billion (10×10^{12}) additions per second, which of the following is the best estimate of the maximum number of cities for which the problem can be solved in 1 hour.

a) 55 cities
b) 18 cities
c) 12 cities

the weight matrix next.

$(V,E), V = \{A, G, H, I\}$, and set of edges E having weights as defined by

a. [_____/3 Points] Trace the execution of the **exhaustive search approach** for the travelling salesman problem described in class on the given instance **assuming that the starting city is *A***. Show the details and arrange your answer in a table.

	A	*G*	*H*	*I*
A	∞	7500	5000	2800
G	7500	∞	3000	6000
H	5000	3000	∞	4000
I	2800	6000	4000	∞

//0.5 point per row: 0.25 for path and 0.25 for cost, final answer for cost is sufficient. If cost is incorrect but the corresponding summation is correct, then we accept the summation and deduct 0.1.

Path	cost
AGHIA	7500+3000+4000+2800=17,300

Path	cost
AGHIA I	7500+3000+4000+2800=17,300
AGIHA	7500+6000+4000+5000=22,500
AHGIA	5000+3000+6000+2800=16,800
AHIGA	5000+4000+6000+7500=22,500
AIGHA	2800+6000+3000+5000=16,800
AIHGA	2800+4000+3000+7500=17,300

	7500=17,300

b. [_____/0.5 Points] What is the cost of the shortest Hamiltonian circuit?
_____16,800_____//to be graded based on student's answer in part a.

c. [_____/0.5 Points] If the exhaustive search algorithm for TSP is programmed on a computer that executes ten billion (10×10^{12}) additions per second, which of the following is the best estimate of the maximum number of cities for which the problem can be solved in 1 hour.

i. 55 cities
ii. 18 cities

Type here to search

executes ten billion (10×10^{12}) additions per second, which of the following is the best estimate of the maximum number of cities for which the problem can be solved in 1 hour.

i. 55 cities
ii. 18 cities
iii. 12 cities

$(n-1)additions\ per\ tour \times (n-1)!\ tours \leq 10^{12} additions\ per\ second \times 3600\ seconds\ per\ hour$

$= 36 \times 10^{15}$

$(55-1) \times (55-1)! = 1.25E{+}73 > 36 \times 10^{15}$

$(18-1) \times (18-1)! = 6.05E{+}15 < 36 \times 10^{15}$

$(12-1) \times (12-1)! = 5.75E{+}9 < 36 \times 10^{15}$

Part 3 [5 Points]

Question 5 [_____/5 Points] A decimal number n (e.g. 5) can be represented as a binary number (e.g. 101). The below algorithm computes the number of bits (digits) in binary representation for a given decimal number n.

```
ALGORITHM BinRec (n)
    if n = 1
        return 1
    else
        return BinRec (⌊n/2⌋) + 1
```

5.1.[_____/1 Points] **Set up** a recurrence relation for the number of times the algorithm's basic operation is executed.

$A(n) =$

$A(1) =$

5.2.[_____/2.5 Point] Solve the recurrence relation using backward substitution, show the details. Find the time complexity.

Hint: use the *smoothness rule* theorem, where: $n = 2^k$.

```
ALGORITHM BinRec (n)
    if n = 1
        return 1
    else
        return BinRec (⌊n/2⌋) + 1
```

I. [_____/1 Points] **Set up** a recurrence relation for the number of times the algorithm's basic operation is executed.

$A(n) = A(n/2) + 1$ 0.5

$A(1) = 0$ 0.5

II. [_____/2.5 Point] **Solve** the recurrence relation using backward substitution, show the details. **Find** the time complexity.

Hint: use the *smoothness rule* theorem, where: $n = 2^k$.

$\Delta(n) = A(n/2) + 1$ 0.5

$\Delta(1) = 0$ 0.5

II. [/2.5 Point] **Solve** the recurrence relation using backward substitution, show the details. **Find** the time complexity.

Hint: use the *smoothness rule* theorem, where: $n = 2^k$.

$\Delta(2^k) = A(2^{k-1}) + 1$ substitute $A(2^{k-1}) = A(2^{k-2}) + 1$ 0.25

$= [\Delta(2^{k-2}) + 1] + 1 = A(2^{k-2}) + 2$ 0.25 substitute $A(2^{k-2}) = A(2^{k-3}) + 1$ 0.25

$= [A(2^{k-3}) + 1] + 2 = A(2^{k-3}) + 3$ 0.25

$= [\Delta(2^{k-1}) + 1] + 2 = \Delta(2^{k-1}) + 3$ 0.25

substitute $\Delta(2^{k-2}) = \Delta(2^{k-3}) + 1$ 0.25

$= \Delta(2^{k-2}) + 2$ 0.25

substitute $\Delta(2^{k-3}) = \Delta(2^{k-4}) + 1$ 0.25

:

$= \Delta(2^{k-i}) + i$ 0.25

:

$= \Delta(2^{k-k}) + k$ 0.25

$= \Delta(1) + k$

$= 0 + k$

$= k$ 0.25

given that $n = 2^k \rightarrow k = \log_2 n$ 0.25

$\Delta(n) = \log_2 n$

$\Delta(n) \in O(\log n)$ 0.5

$A(n) \in O(\log n)$ 0.5

III. [_____/1.5 Point] **Solve** the recurrence relation using the Master theorem (given below), show the details. **Find** the time complexity.

$a = 1,\ b = 2,\ d = 0$ 0.25 (0.75 each)

$a = b^d$ -> case (2) 0.25

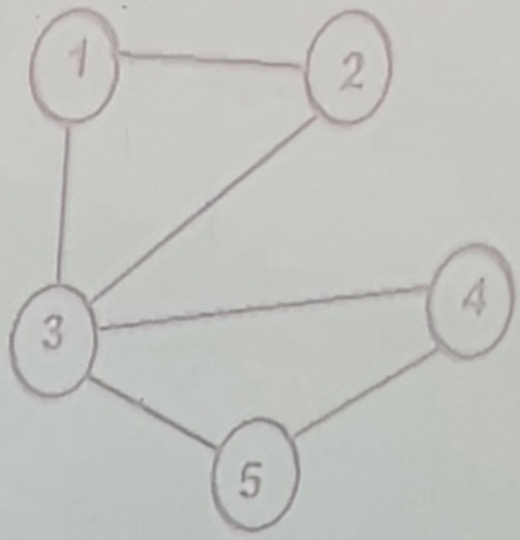
$\Delta(n) = \Theta(n^0 \log n)$

$$T(n) \in \begin{cases} \theta(n^d) & \text{if } a < b^d \\ \theta(n^d \log n) & \text{if } a = b^d \\ \theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

$A(n) \in \Theta(\log n)$ 0.5

Part 4 [6 Points]

Question 6 [_____/3 Points]: Consider the *clique* problem: given an undirected graph G and a positive integer k, find all cliques of size k in G, if any exist, or determine that none exist. i.e., find all complete subgraphs of G having k vertices. For example, in the graph shown to the right, $n = 5$. $G = (V,E)$, $V = \{1,2,3,4,5\}$, $E = \{(1,2),(1,3),(2,3),(3,4),(3,5),(4,5)\}$, there are two cliques of size $k = 3$: namely the clique C_1 with vertices $\{1,2,3\}$ because every pair of vertices in C_1 are adjacent. Similarly, the clique C_2 with the vertices $\{3,4,5\}$. However, there are no cliques of size $k = 4$.



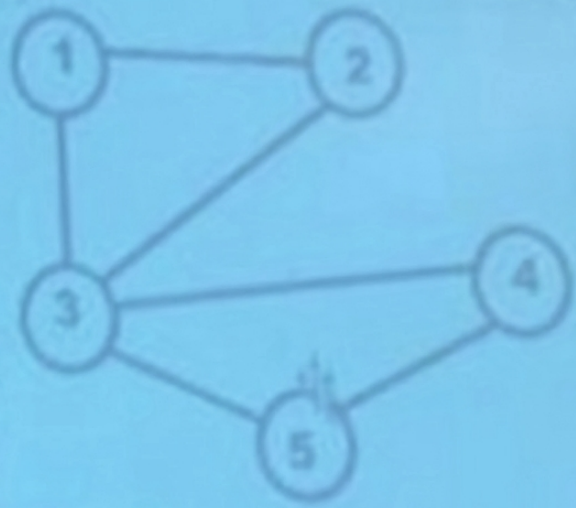
6.1. [____/1 Points] Give a *high-level description* of an exhaustive-search algorithm for the clique problem: find and output all cliques of size k in a graph, $G(V,E)$, of size n , or determine that none exist.

6.2. [____/1 Point] Now consider the *maximum clique problem*: find *a clique* of *the maximal size* in G. In the above instance, the maximal clique size is 3, since there are no cliques of larger size in this instance. Note that this time, the clique size k is unknown. Give *a high-level description* of an exhaustive-search algorithm for the maximum clique problem.

6.3. [____/0.5 Point] What is the computational complexity of your algorithm for Question 5.b in big-O?

6.4.[____/0.5 Point] Suggest a way for optimizing the algorithm?
Hint: think of excluding from the initial search vertices that could never be part of a clique of size k.

Question 5.1 [5 Points]: Consider the *clique* problem: given an undirected graph G and a positive integer k, find all cliques of size k in G, if any exist, or determine that none exist. i.e., find all complete subgraphs of G having k vertices. For example, in the graph shown to the right, $n = 5$, $G = (V, E)$, $V = \{1,2,3,4,5\}$, $E = \{(1,2),(1,3),(2,3),(3,4),(3,5),(4,5)\}$, there are two cliques of size $k = 3$: namely the clique C_1 with vertices $\{1,2,3\}$ because every pair of vertices in C_1 are adjacent. Similarly, the clique C_2 with the vertices $\{3,4,5\}$. However, there are no cliques of size $k = 4$.



a. [_____/1 Points] Give a *high-level description* of an exhaustive-search algorithm for the clique problem: find and output all cliques of size k in a graph, $G(V, E)$, of size n, or determine that none exist.

For all subsets of n of size k vertices [0.25 for all; 0.25 subsets of size k]
1. Check whether every pair of vertices in the subset is connected by an edge (clique). [0.25 for checking for each pair; 0.25 for output of cliques]
2. If the subset is a clique then output the subset

b. [____/1 Point] Now consider the *maximum clique problem*: find *a clique* of *the maximal size* in G. In the above instance, the maximal clique size is 3, since there are no cliques of larger size in this instance. Note that this time, the clique size k is unknown. Give *a high-level description* of an exhaustive-search algorithm for the maximum clique problem.

For $k = 1$ *to* n do [0.25 point]
For all subsets of n of size k vertices [0.25 point]
1. Check whether every pair of vertices in the subset is connected by an edge (clique). [0.25 points]
2. If the subset is a clique then save the subset [0.25 points]

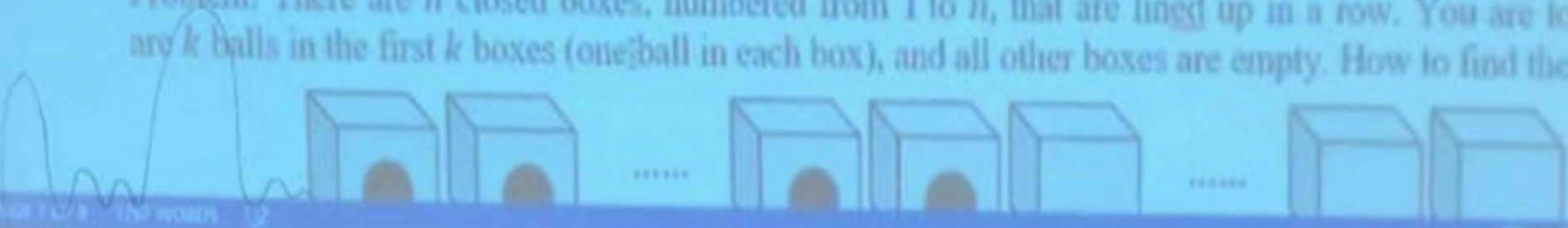
Output the last (largest) subset

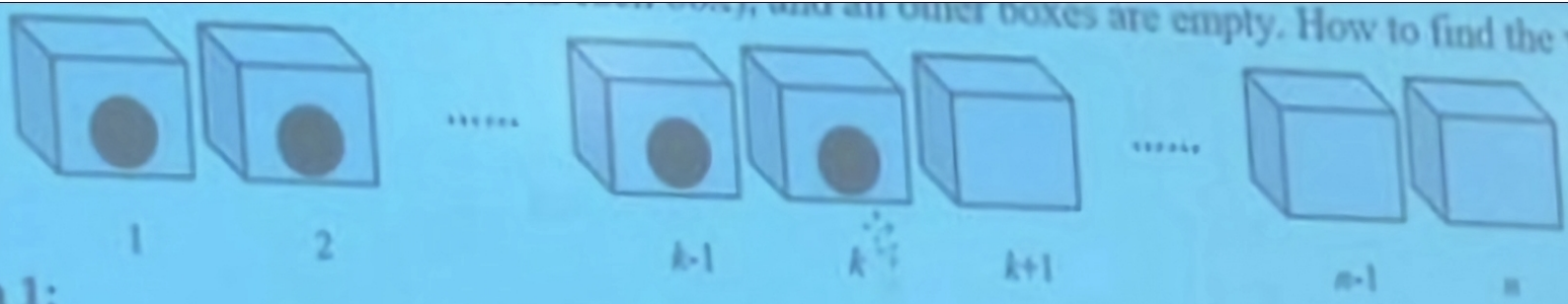
c. [____/0.5 Point] What is the computational complexity of your algorithm for Question 5.b in big-*O*?
$O(2^n)$

d. [____/0.5 Point] Suggest a way for optimizing the algorithm?
Hint: think of excluding from the initial search vertices that could never be part of a clique of size k.
Exclude vertices having degree less than k.

Question 5.2 [3 Points]: Consider the problem below, and suggest TWO algorithm design techniques and give a high-level description of the TWO algorithms to solve it. For each algorithm, name the used algorithm design technique and state its complexity.

Problem: There are n closed boxes, numbered from 1 to n, that are lined up in a row. You are told that there are k balls in the first k boxes (one ball in each box), and all other boxes are empty. How to find the value of k?





Algorithm 1:

Algorithm design technique: _____[brute force algorithm] //0.5

High-level description of the algorithm: //0.5

Scan the n boxes from the last down to the first (or vise versa) checking for a ball. Output the index of the first box found containing a ball (or the index of the first empty box -1 if scan is from 1 to n)

Algorithm 2:

Algorithm design technique: I [D&C] //0.5

High-level description of the algorithm: //0.5

similar to binary search in a list of n boxes, with the following mappings:

empty box: check previous. If full return its index else binary-search the left half of list
full box: check next box. If empty return the full boxes index else binary search the right half of the list